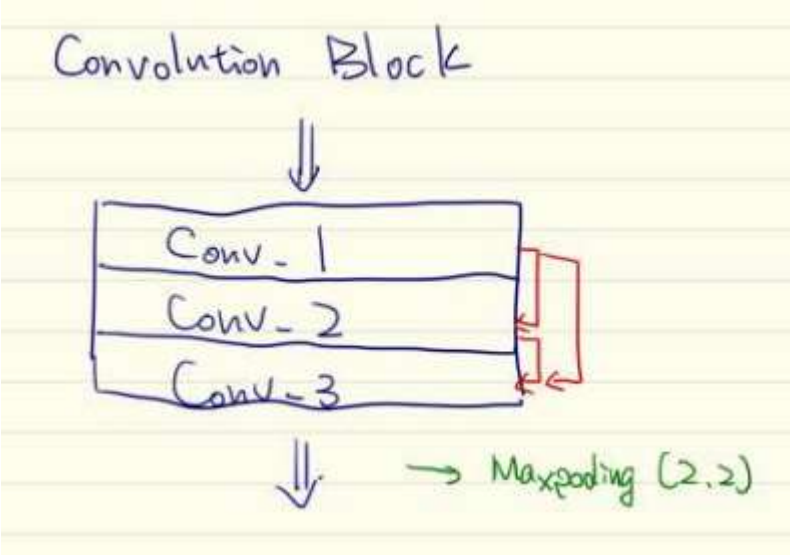
HW4_Report　R06525087 郭漢遜

（一）Model description + improve your performance：

我所採用的模型結構是 ACGAN，也就是比起原本的 GAN，生成器除了接收隨機向量，同時也接受一組代表指定分類條件的向量（我用長度 12 及 11 的一維向量代表髮色及眼睛顏色），而判別器在判別每一張圖片真假時，也同樣會收到同樣的向量作為提示。

因此我的模型都會有三個 Input Layer，對於生成器就是（〔隨機向量，髮色向量，眼睛顏色向量〕），對於判別器就是（〔真/假的圖片，髮色向量，眼睛顏色向量〕）。

而模型的卷積核心構造如下：



在判別器及生成器中，都採用了多層卷積層，而卷積層是以三層為一單位，input 會經過三個結構一模一樣的卷積層，而第一個卷積的輸出會輸出到下一層及下下層，這樣在深度多層卷積的情況下，可以避免梯度消失及保留圖片資訊，讓模型有更多的訓練空間。而且在許多利用卷積的製作高清晰影像的研究中（如 SrrGan）都採用了類似的方法，因此想要在這次作業中試驗看看。

不過可能也由於可學習空間太大，現在時間已經訓練的１千五百個 epoch，雙方都還停留在每次能收斂到準確率９５％以上（後面紀錄會說明），相信如果有更多時間讓他們收斂到五十五十的話，會有很棒的圖片。

◎判別器結構：

Input => 4 Convolution Blocks => Flatten ,
（Flatten＋Hair_Vector＋Eyes_Vector） => Dense(output_units = 1)

對於判別器而言，輸入是一張（96,96,3）的圖片並先經過四個卷積組（一個卷積組就是上面圖示的三層＋MAXPOOLING），採用的 KERNEL 大小都是 3 乘 3，有 Padding，通道數分別為 32, 64, 128 ,256, 最後攤平成一維向量後，此時才把頭髮及眼睛顏色的向量與它接起來再輸入到 1 個數字，當然以 sigmoid 輸出．

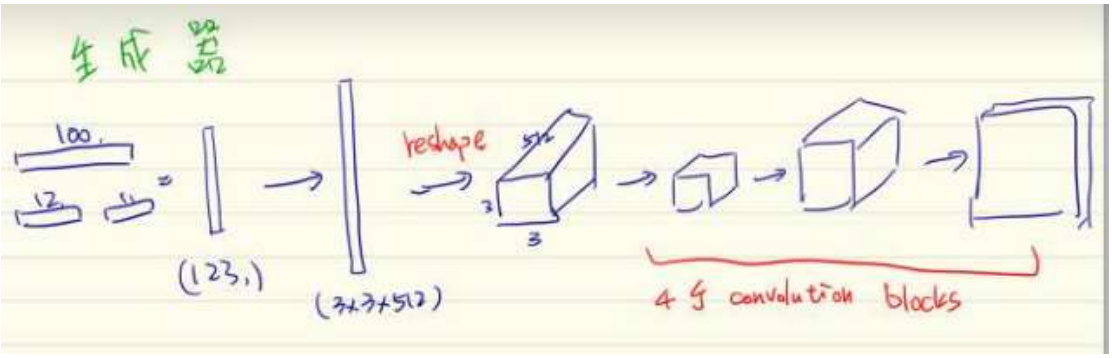
◎生成器結構：

Input =（noise＋Hair_Vector＋Eyes_Vector） ,
⇨ Dense(3*3*512) => Reshape(3,3,512) => 4 Convolutuin Blocks
⇨ 最後一層卷積為 3 個 filter

生成器的結構比較麻煩一點，輸入首先有個隨機生成的 100, 向量，限制在 1 到-1 之間，然後再看這張圖要訓練生成甚麼條件的圖片，在輸入時就接上長度 11 及 12 的 one hot 向量（這裡的向量使用與判別器完全相同），然後經過全連接層變成 3*3*512 長度，所光這層的參數就有（3*3*512）*（100+11+12），乘起來不加 BIAS 總共一百多萬個了！！基本上生成器跟判別器都有同樣的情況，全連結層的參數都超肥大！（後來因為）

到這邊已經可以視為一個 3 乘 3 大小，通道為 512 的圖片，然後同樣經過四個卷積 BLOCK，BLOCK 結構與上面是相同的只是通道是逐漸除以二變少，另外每個 BLOCK 的第一個卷積層必須使用轉置卷積，Strides 設定為 2 使得移動兩步才會經過一個像素點（在實作上就是把輸入的圖片像素與像素之間空出一格以讓圖片變大），其他部分跟一般卷積一樣．

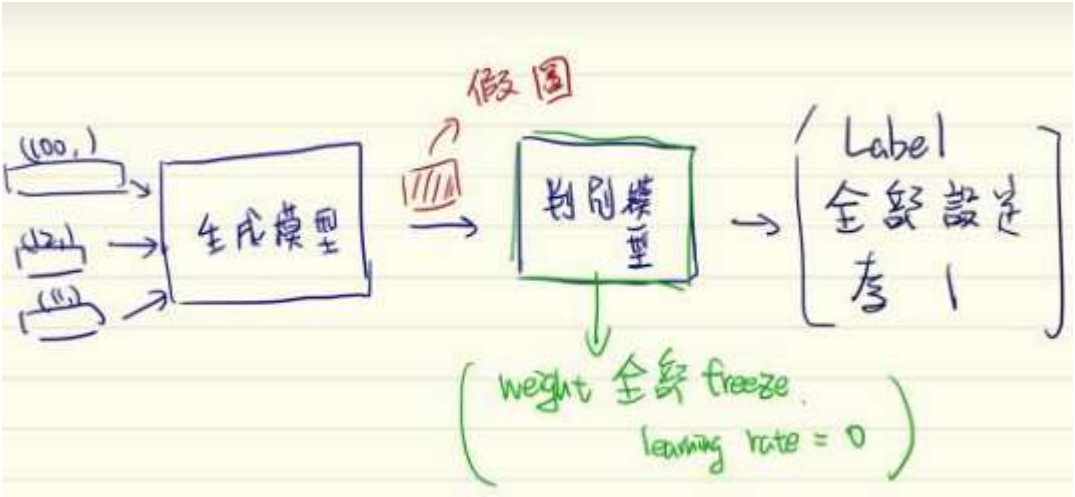
另外在第四個卷積 BLOCK，轉置的部分 strides 是設定為 4 的，於是經過四層就能從(3,3)變為(96,96)，另外在卷積的最後一層輸出當然是 3 通道，及 sigmoid．



（二）Experiment settings and observation：

在 Training 的時候，判別器的部分比較正常，就是一組真圖片，一組 G_MODEL 用 Predict 產生的假圖片，輸出就是真圖片的 LABEL 為 1，假圖片為 0 ·

生成器比較巧妙一點，訓練時其實是把生成器跟判別器黏接在一起的大模型



所以送到ＧＰＵ裡面的要送兩個模型，生成器的部分不斷產生假圖，直接送到判別器裡面並希望最後能夠輸出１，這樣很方便，但是會導致訓練有限制就是模型不能開太大Ｑ＿Ｑ，如果分開來操作在硬體上就能更活用，要怪就怪我的 1050 顯示卡太爛了記憶體只有２ＧＢ·因此Ｇ_model 的訓練實際上同時 work 了兩個模型~~~~

Training 設定：

	DATA 數量	EPOCH	優化	LEARNING_RATE	BATCH_SIZE
D_model	分辨 22836	1	ADAM	0.0001	32
G_model	生成 11418	1	ADAM	0.0001	32

GAN 的交替訓練在一開始時，G_MODEL 跟 D_MODEL 都在每一個 epoch 都會非常快收斂到準確率 = 1，也就是只要少少的 DATA 就可以快速騙過／成功分辨。

但是隨著訓練次數增加，它們往１收斂的速度會越來越慢，但是可能由於我 LEARNING RATE 只有 0.0001 的關係，一直到各 1000 個 EPOCH 的時候，兩個 MODEL 在每一個 EPOCH 都還是能達到接近 1 的準確率。

只是可能原本在一個 EPOCH 的前段時候就能夠到 1，到後面必須要快要整個 DATA 看完才能到 1，也就是雙方能力都增強了，對方需要更多的 DATA 才能破解對方。而 D_MODEL 幾乎每次都能 TRAIN 到 0.99 以上，而 G_model 最後大概下降到 0.95 或 0.96 左右。

在 TRAIN 完 1 千次之後其實從結果來看方向是正確的，只是速度實在是太慢了會來不及交作業，所以決定在這時候做 LEARNING RATE 的調整，但是發現比如直接調整到 0.0005 甚至 0.001，也就是原本的五倍以上的時候，準確率會直接壞掉。

就是 D_MODEL 直接變成 0.5 而 G_MODEL 變成 1，也就是分辨器已經無法學習真假，這樣也導致 G_MODEL 停止學習。推測是過大的 LEARNING RATE 導致往不對的方向優化時，模型整個回不來了！！

因為同時把 LEARNING RATE 開大會導致上面的結果，因此決定先維持 D_MODEL 的 LEARNING RATE 為 0.0001，而把 G_MODEL 的 LEARNING RATE 調整為三倍也就是 0.0003。會這樣做也是因為在記錄中，G_MODEL 的 LOSS 跟 ACC 的確是約略低於 D_MODEL 的。

這樣的方式，在 1067 次的時候突然就停止了，也就是 D_model 準確率 0.5 而 G_model 準確率 1。學習無法繼續！

因此前幾天把模型重新從 1000 按照原始設定放下去繼續訓練，目前（禮拜六）到 1500 次，還是處於可以有效訓練但是每次都會完全打敗對方的程度，可能在作業期限前無法收斂，但是是可以畫出能辨認的人物圖的，以上是我目前的研究成果@@！